AMENDMENTS TO THE CLAIMS:

1. (Currently amended) The use of multiple threads in association with a network processor and accessible data available in a tree search structure, including the steps of:

a) providing multiple instruction execution threads as independent processes in a sequential time frame;

b) encoding a processor instruction to select an event selected from the group consisting of either a short latency or a long latency event;

~~b)~~ c) queuing the multiple execution threads to have overlapping access to the accessible data available in said tree search structure;

~~c)~~ d) executing a first thread in a queue;

~~d)~~ e) transferring control of the execution to the next thread in the queue upon the occurrence of an event that causes execution of the first thread to stall; and

~~e)~~ f) when the stall is due to a short latency event, returning control to the first thread when the event is completed, and when the stall is due to a long latency event, retaining full control by the next thread until the next thread becomes blocked.

2. (Canceled)

3. (Currently amended) The use of multiple threads according to claim 1 wherein a the processor instruction is encoded to select a the short latency event.

4. (Canceled)

5. (Currently amended) The use of multiple threads according to claim 1 wherein a the processor instruction is encoded to select a the long latency event.

6. (Canceled)

7. (Original) The use of the multiple threads according to claim 1 wherein the threads have overlapping access to shared remote storage via a pipelined coprocessor by operating within different phases of a pipeline of the coprocessor.

8 – 10 (Canceled)

11. (Currently amended) A network processor that uses multiple threads to access data available in a tree search structure, including:

a) a CPU configured with multiple instruction execution threads as independent processes in a sequential time frame;

b) a thread execution control for

1) queuing the multiple execution threads to have overlapping access to the data to be accessed;

2) executing a first thread in the queue;

3) encoding a processor instruction to select an event selected from the group consisting of either a short latency event or a long latency event;

and

4) transferring control of the execution to the next thread in the queue upon the occurrence of an event that causes execution of the first thread to stall, said thread execution control including control logic responsive to the type of event causing the execution to stall that a) temporarily transfers control to the next thread when execution stalls due to[a] the short latency event and returns control to the first thread when the short latency event is completed, and b) transfers full control of the execution to the next thread when execution of the first thread stalls due to [a] the long latency event with execution continuing on the next thread even after the long latency event is completed, until the next thread becomes blocked.

12. (Canceled)

13. (Currently amended) The processor according to claim 11 wherein a the processor instruction is encoded to select a the short latency event.

14. (Canceled)

15. (Currently amended) The processor according to claim 11 wherein a the processor instruction is encoded to select a the long latency event.

16. (Canceled)

17. (Currently amended) The processor according to claim ~~16~~ 11 wherein the threads have overlapping access to shared remote storage via a pipelined coprocessor by operating within different phases of a pipeline of the coprocessor.

18. (Canceled)

19. (Currently amended) The processor according to claim 11 wherein the processor uses zero overhead to switch execution from one thread to the next by giving each thread access wherein the general purpose registers and the local data storage are made available to the processor by providing one address bit under the control of the thread execution control logic and by providing the remaining address bits under the control of the processor.

20 - - 21 (Canceled)

22. (Currently amended) The processor according to claim ~~20~~ 19 wherein the processor is capable of simultaneously addressing multiple register arrays, and the thread execution control logic includes a selector to select which array will be delivered to the processor for a given thread.

23. (Currently amended) The processor according to claim ~~20~~ 19 wherein the local data storage is fully addressable by the processor, an index register is contained within the register array, and the thread execution control has no address control over the local data storage or the register arrays.

24 - 35 (Canceled)